WIFI , BLUETOOTH WITH SEVEN SEGMENT UGV

Future Wireless Communication-IITH
FWC220107 S SRIKANTH REDDY

Contents

1 Components

Component	Value	Quantity
Vaman Board		1
USB-UART		1
UGV Chasis		1
DC Motors		2
Motor Driver Unit		1
Jumper Wires	F-F	10
Breadboard		1

Table 1.0

2 Circuit Connections

Make the Circuit Connections as per the table below.

Vaman Board	Motor Driver Unit
Pin 21	Right Motor Input 1
Pin 18	Right Motor Input 2
Pin 23	Left Motor Input 1
Pin 22	Left Motor Input 2
5V	VCC
GND	GND

Table 2.0

3 L293 Motor Driver

Make the Motor Driver Connections as per the table below.

INPUT	VAMAN BOAD	OUTPUT	MOTOR
A1	PYGMY 21	Vcc	5V
A2	PYGMY 18	GND	GND
EN	-	MA1	MOTOR A1
VCC	5V	MA2	MOTOR A2
B2	PYGMY 23	MB1	MOTOR B1
B1	PYGMY 22	MB2	MOTOR B2
5V	VCC	-	-
GND	GND	-	-

Table 3.0

4 WIFI CAR Connections

Make the Circuit Connections as per the table below.

Vaman Board ESP 32	Motor Driver Unit
Pin 16	Right Motor Input 1
Pin 17	Right Motor Input 2
Pin 18	Left Motor Input 1
Pin 19	Left Motor Input 2
5V	VCC
GND	GND

Table 4.0

5 UART Connections

Connect the USB-UART pins to the Vaman ESP32 pins according to Table.

Vaman LC PINS	UART PINS
GND	GND
ENB	ENB
TXD0	RXD
RXD0	TXD
O	IO 0
5V	5V

Table 5.0

6 SevenSegment

All codes used in this document are available at the following link.

https://github.com/gadepall/ugv/tree/main/codes/sevenseg

7 Seven Segment Connections

Seven Segment to the Vaman board according to the given connection given in the table

VAMAN PINS	SEVEN SEGMENT PINS
IO -32	a
IO-33	b
IO-25	c
IO-26	d
IO-27	e
IO-14	f
IO-12	g

Table 6.0

Figure 1 - VAMAN BOARD

8 Working

8.1 Hardware Level:

On the hardware level there are three key points: SPI, Wishbone Interfacing and Address Mapping.

On the Vaman Board, we have an EOS S3 and ESP32. The Communication between these two happens via SPI i.e, **Serial Peripheral Interface**. And this is facilitated only when all the 4 jumpers on the board are closed.

The EOS S3 has an ARM M4 Core, a FPGA unit and 512 KB of SRAM along with a AHB. The SRAM is divided into 4 banks. Banks 0-2 are accessible only by the ARM M4. Bank 3 is accessible by any master connected to **Always-ON Bus**. FPGA a slave on that AON Bus.
The same can be observed from the figure below.

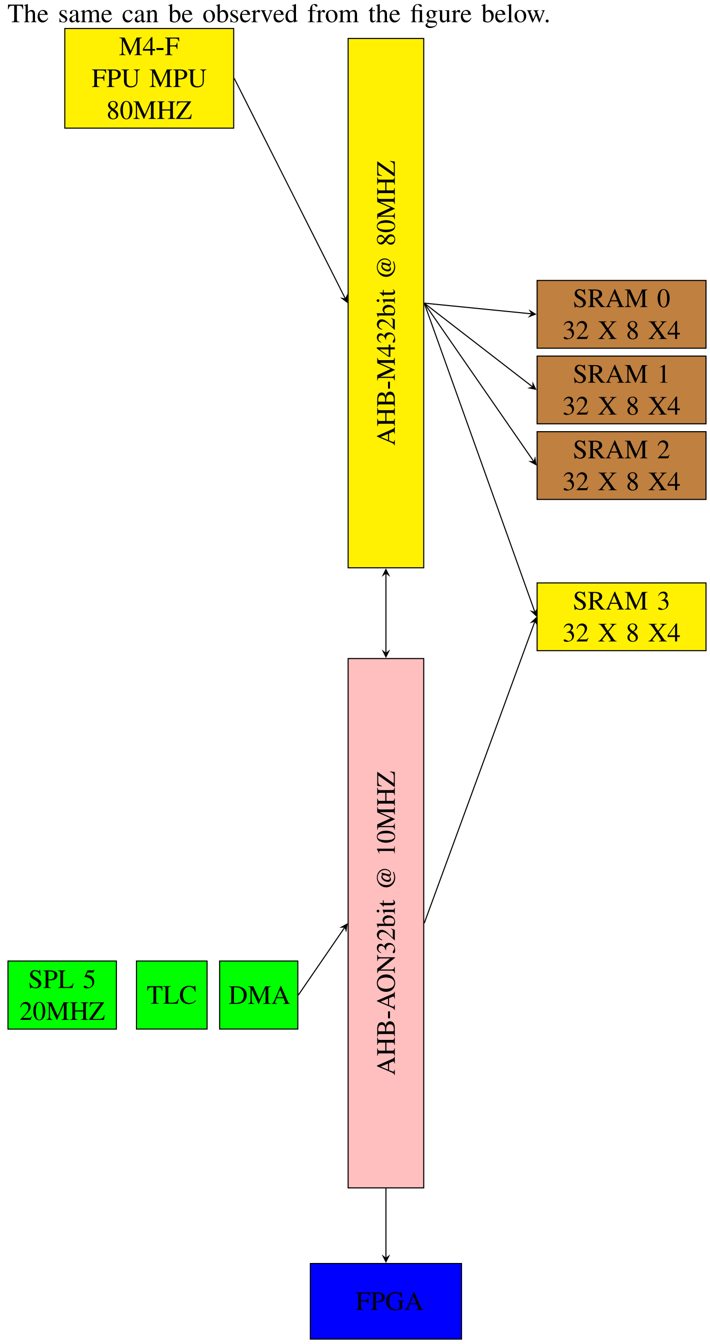


Figure 2 - EOS S3 Architecture

Now, the communication between ARM M4 Core and the FPGA unit takes place via the AHB Bus. For this to happen, we implement **Wishbone slave interface** on the FPGA. Without this Wishbone slave interface the communication with FPGA registers is not possible. The master requests from the ARM4 Core on the AHB are converted to wishbone signals and hence reach FPGA.

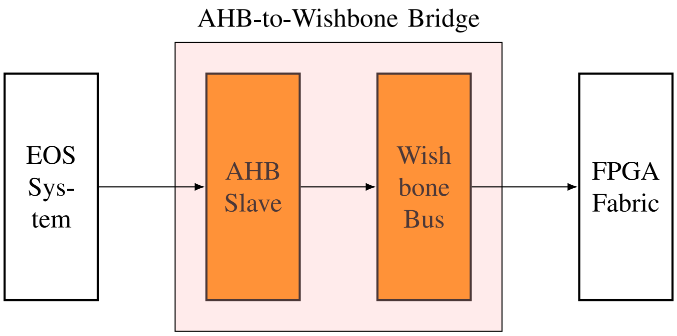


Figure 2 - Wishbone Slave Interface

And the next key thing is to know the starting address of the FPGA Registers in EOS S3 Memory organisation (which is provided by the manufacturer). Hence, mapping them to read or write the FPGA Registers.

8.2 Code:

In the code also there are three major processes that take place in ESP32, ARM Core and FPGA.

Firstly, The ESP32 collects the joystick movement data from the Dabble app connected via bluetooth. It receives the x,y co-ordinates in the range [-7,7]. EP32 then scales it to [0,255] and places it in the Arrays declared in the ARM Core.

The ARM Core now has the joystick movement data in its 8 bit registers. Which it passes on to the mapped FPGA Registers via the AHB.

Then in FPGA we implement the Wishbone slave interface to read the data sent by the ARM Core via AHB. Now in the FPGA Unit, the Pulse Width Modulation takes place. The corresponding PWM values for the joystick movement data are generated and sent back to the ARM Core.

Now the ARM Core running a loop, Checks if the Cross or Square button is pressed and also reads these changes in the PWM values. It then sends signal to the DC Motors of the UGV to rotate the wheels accordingly.

9 Execution

1. Download the repository

```
svn co https://github.com/srikanth9515/FWC/tree/main/UGV
```

2. Build the ESP32 firmware

```
cd esp32_pwmctrl
pio run
```

3. Flash ESP32 firmware (connect USB-UART adapter)

```
pio run −t nobuild −t upload
```

4. If using termux, send .pio/build/esp32doit-devkit-v1/firmware.bin to PC using

```
scp .pio/build/esp32doit−devkit−v1/firmware.bin
    Username@IPAddress:
```

5. Modify line 140 of config.mk to setup path to pygmy-sdk and then Build m4 firmware using

```
cd m4_pwmctrl/GCC_Project
make
```

6. If using termux, send output/m4_pwmctrl.bin to PC using

```
scp output/m4_pwmctrl.bin username@IPaddress:
```

7. Build fpga source (.bin file)

```
cd fpga_pwmctrl/rtl
ql_symbiflow −compile −d ql−eos−s3 −P pu64 −v ∗.v −t
    AL4S3B_FPGA_Top −p quickfeather.pcf −dump jlink
    binary
```

8. If using termux, send AL4S3B_FPGA_Top.bin to PC using

```
scp AL4S3B_FPGA_Top.bin username@IPaddress:
```

9. Connect usb cable to vaman board and Flash eos s3 soc, using

```
sudo python3 <Type path to tiny fpga programmer
    application> −−port /dev/ttyACM0 −−appfpga
    AL4S3B_FPGA_Top.bin −−m4app m4_pwmctrl.bin
    −−mode m4−fpga −−reset
```

10. Install the **Dabble app** on the Mobile from the **Playstore**. Connect it to the **ESP32** on the Vaman Board using **Bluetooth**. Change the controls to **Joystick mode** to navigate the UGV.

10 Execution For WIFI UGV

1. Download the repository

```
https://github.com/srikanth9515/FWC/tree/main/WIFI_UGV
```

2. Build the ESP32 firmware

```
CD WIFI_UGV
pio run
```

3. Flash ESP32 firmware (connect USB-UART adapter)

```
pio run −t upload
```

4. Connect your own TAB /Phone Hot spot and Enter Your SSID Password

```
const char∗ ssid = ”srikanth”; /∗Enter Your SSID∗/
const char∗ password = ”srikanth123”; /∗Enter Your
    Password∗
```

5. Install the **wifi ugv Toy car APK app** on the Mobile from the **Playstore**. Connect it to the **ESP32** on the Vaman Board using **Wifi**. Change the controls to **Joystick mode** to navigate the UGV.

11 Components and Specifications

Figure 3 - DC motors

Figure 4 - UGV frame/chassis

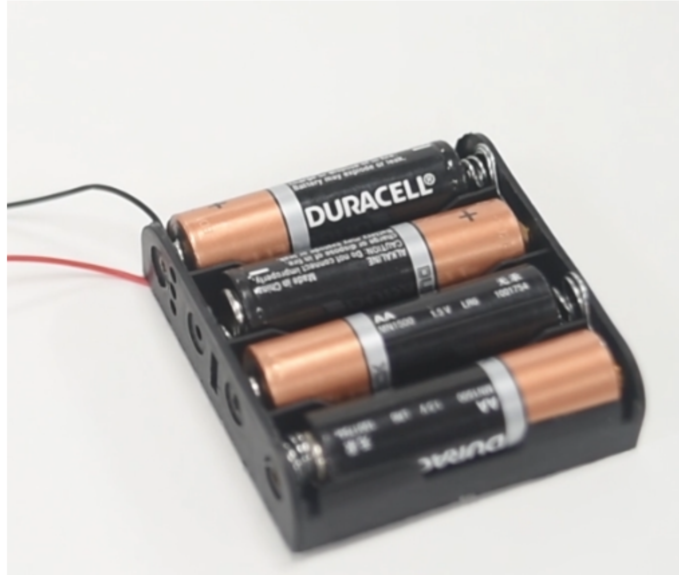
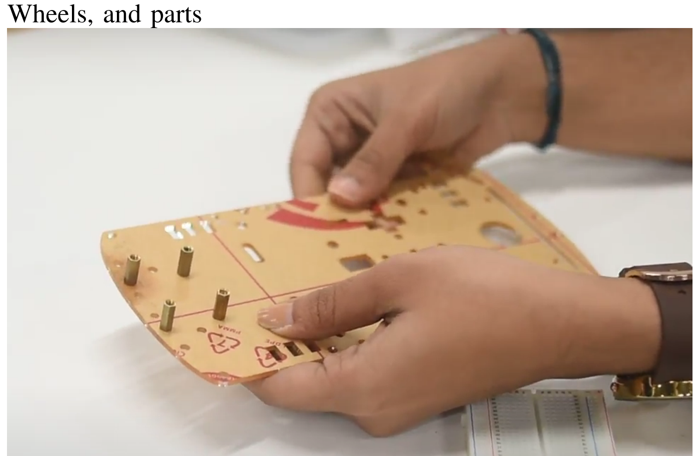
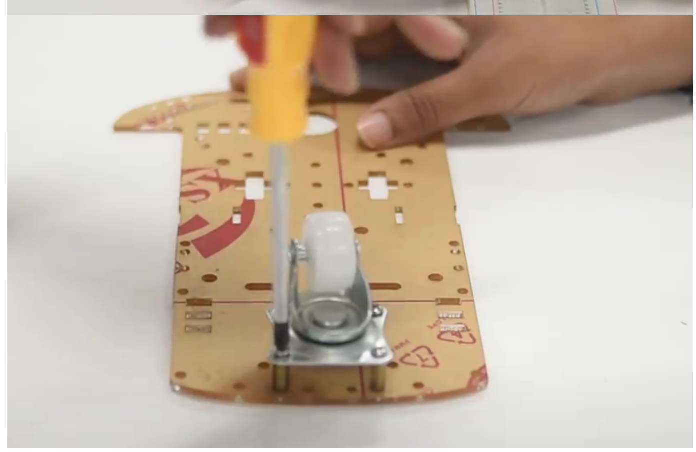


Figure 5 - Batteries

Assemble the Chassis using the provided nuts/screws, Wheels, and parts





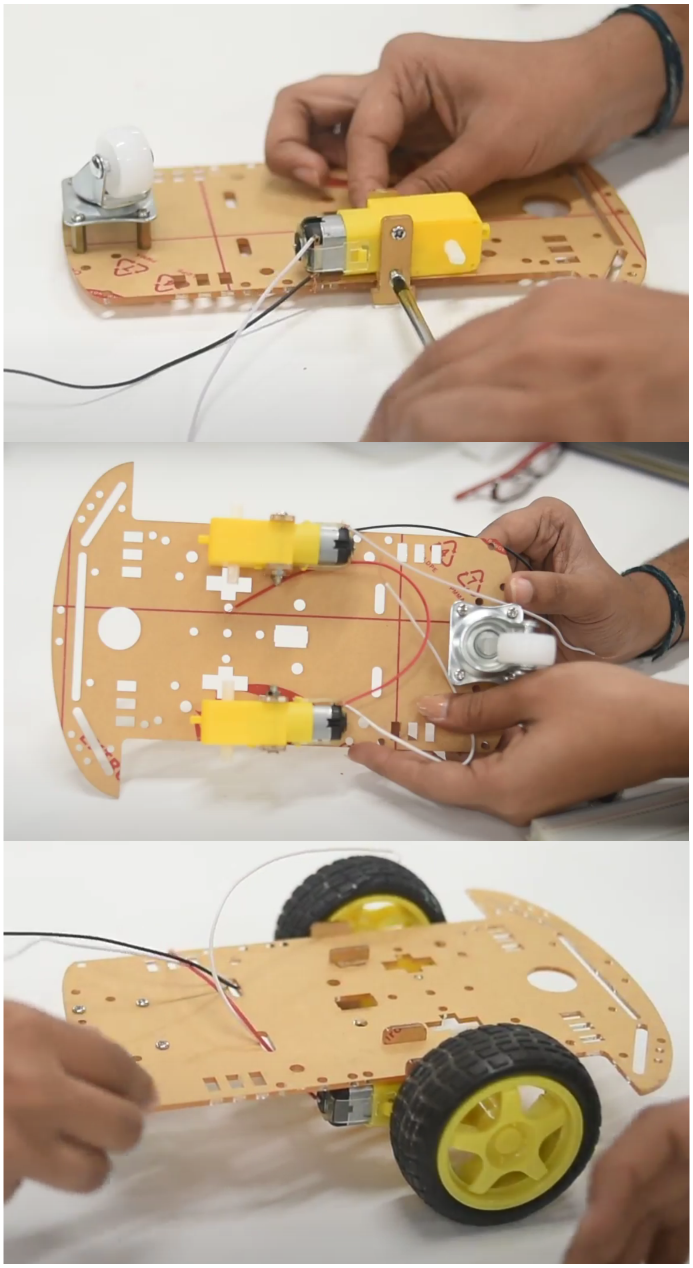


Figure 6 - Assembling the UGV kit

◦ Fix the Vaman controller and ESP32 on the chassis.
◦ Fix the Dual motor driver IC along with a small breadboard on the chassis.
◦ Fix the Li-Po battery on the chassis and insert AA batteries in the battery holder.

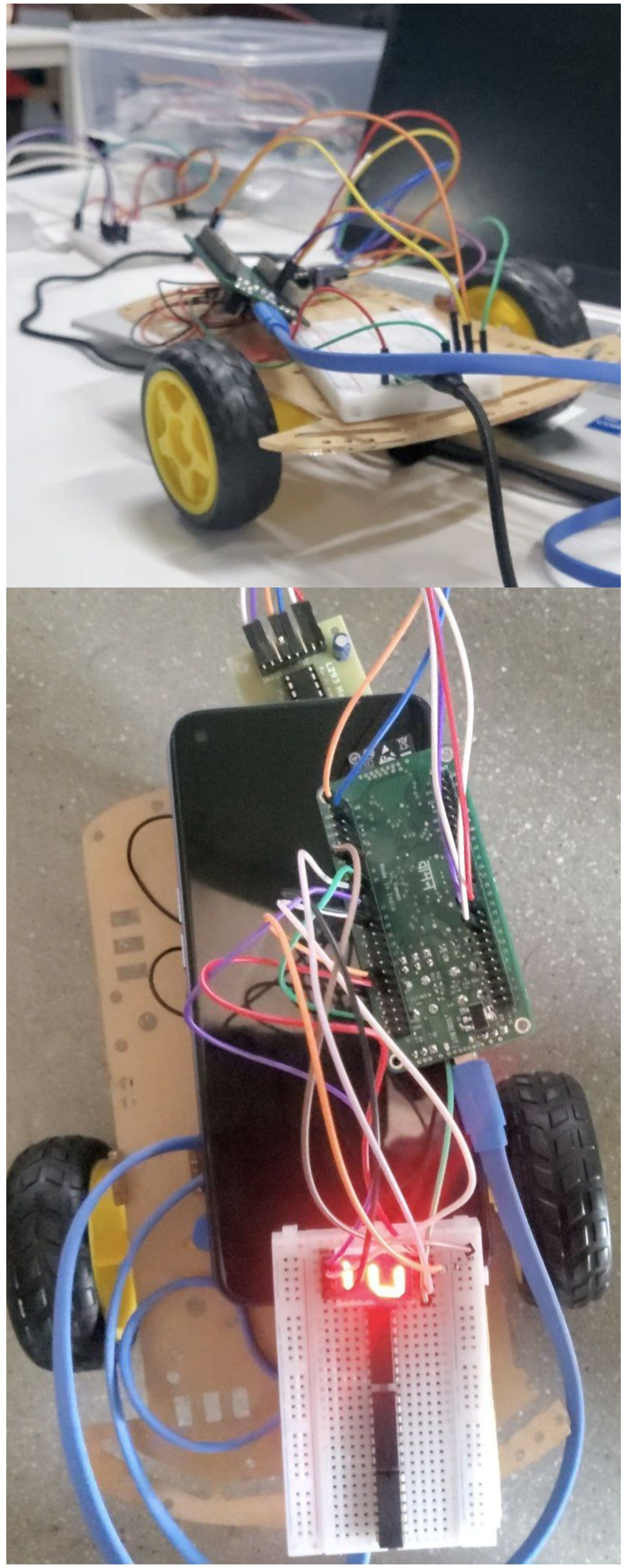


Figure 7 - Connections UGV

*Connect the battery supply and turn on the power to various equipment.

*Download the “dabble” application from the play store on an Android phone.
*Using dabble application, connect to the ESP32 on the UGV kit using Bluetooth connection.
* Control the navigation of the UGV kit using the GUI controls on the dabble application.

12 Seven Segment Display Output For WIFI

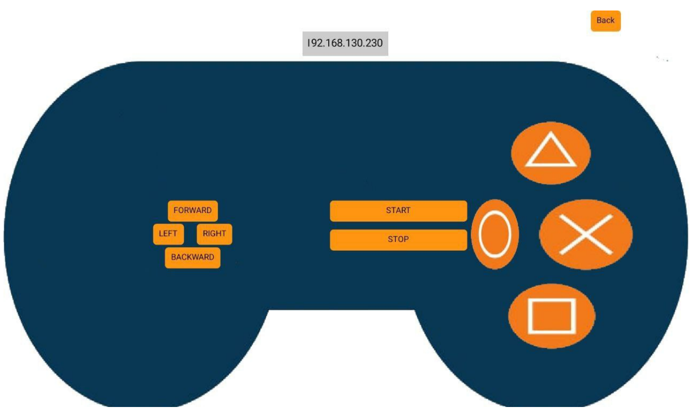


Figure 8 - Seven segment Display with UGV

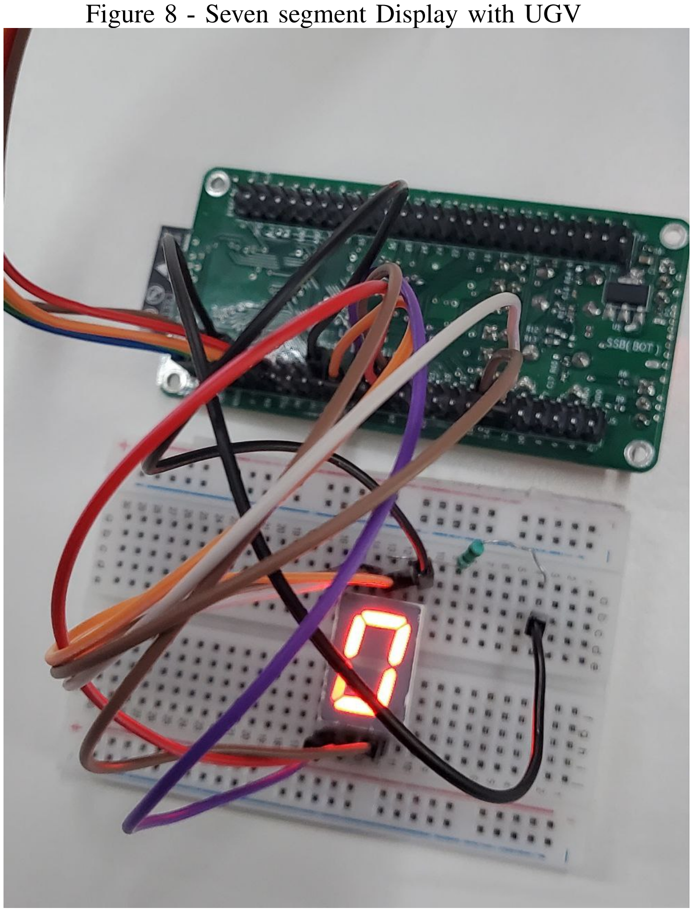


Figure 9 - Stop Will Display zero

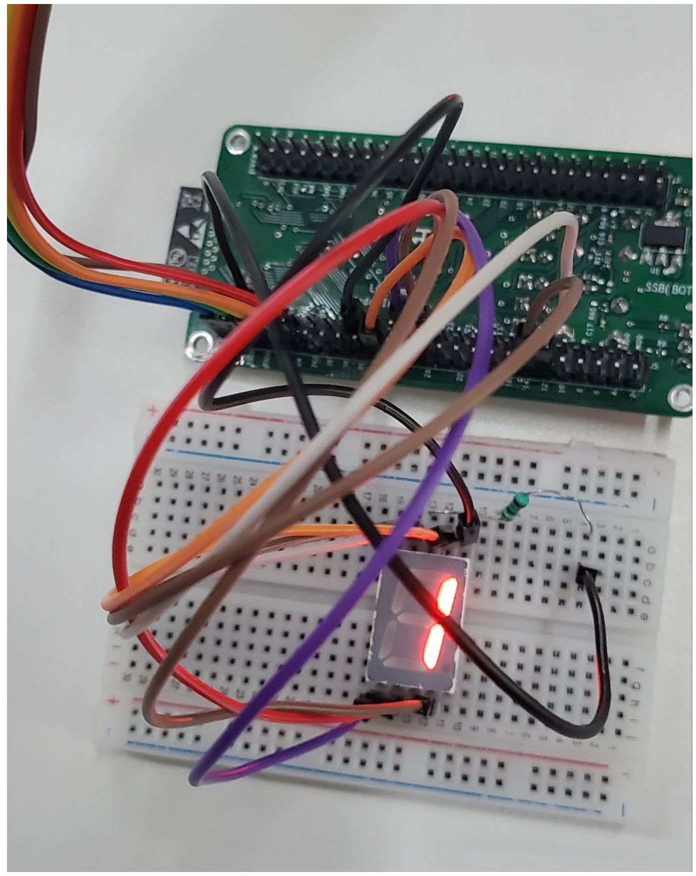


Figure 10 - Start Will Display One

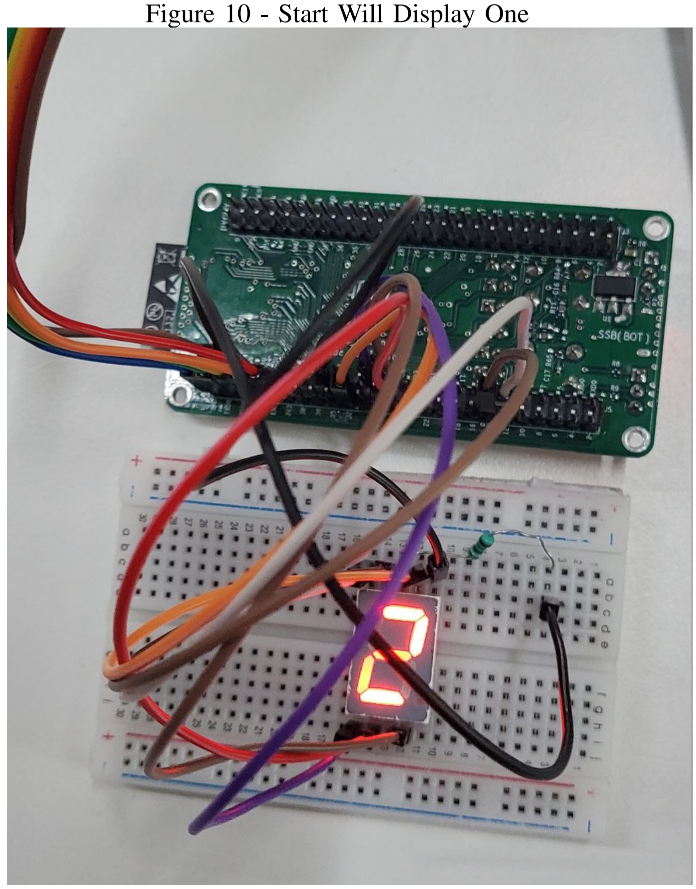


Figure 11 - Forward Will Display Two

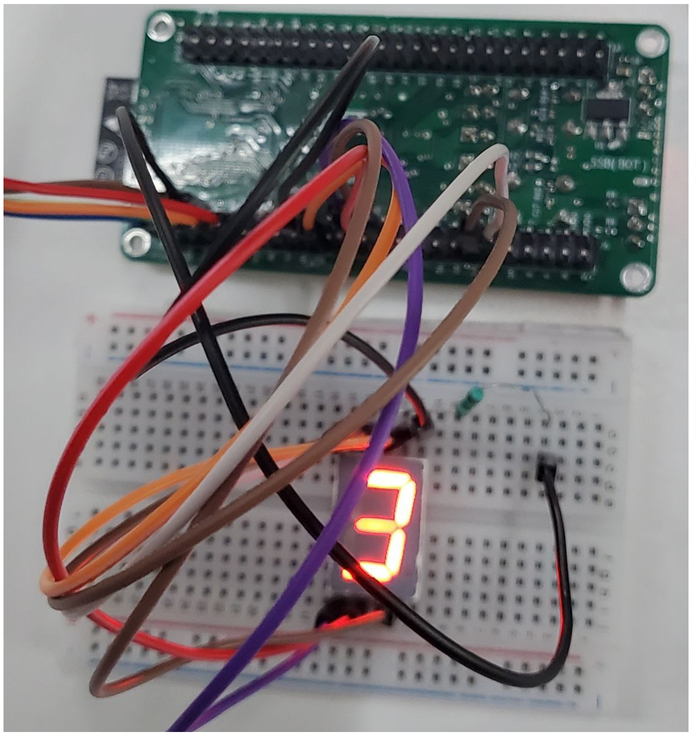


Figure 12 - Back ward Will Display Three

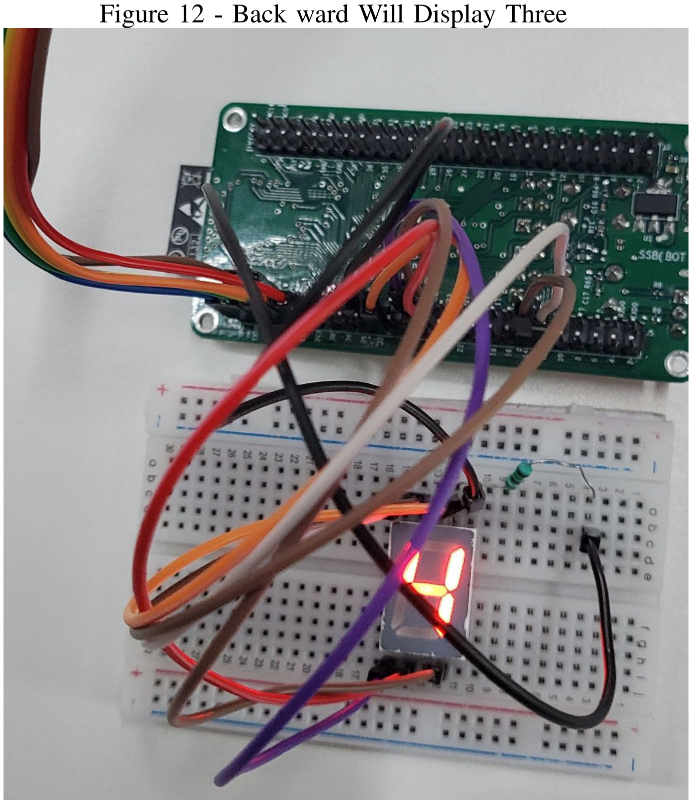


Figure 13 - Left Will Display Four

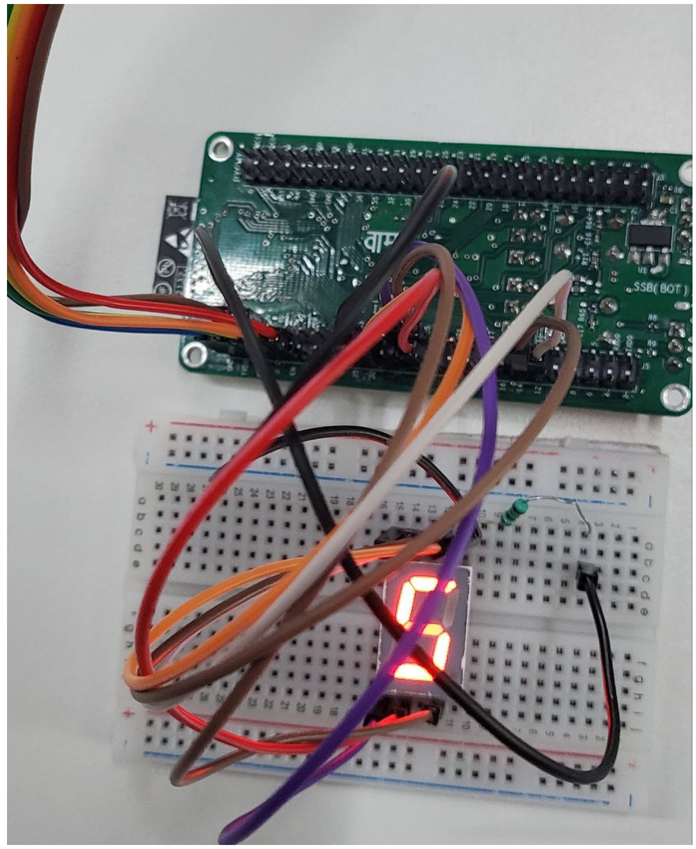


Figure 14 - Right Will Display Five

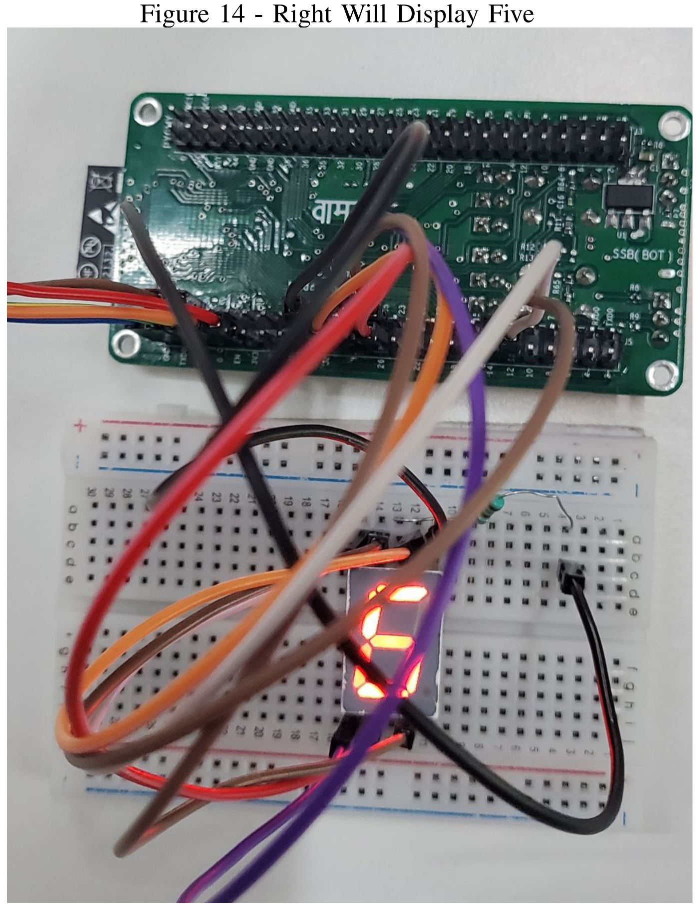


Figure 15 - Cross Will Display Six

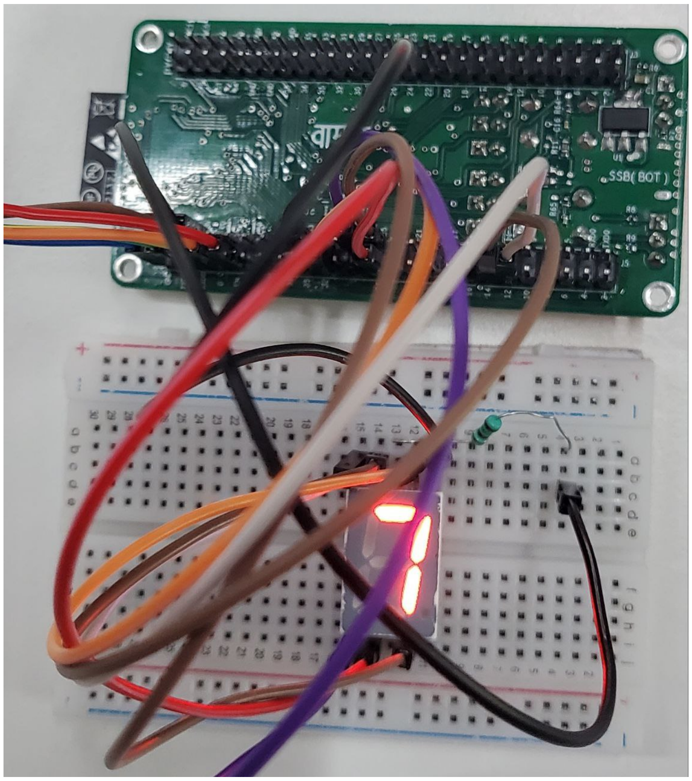


Figure 16 - Circle Will Display Seven

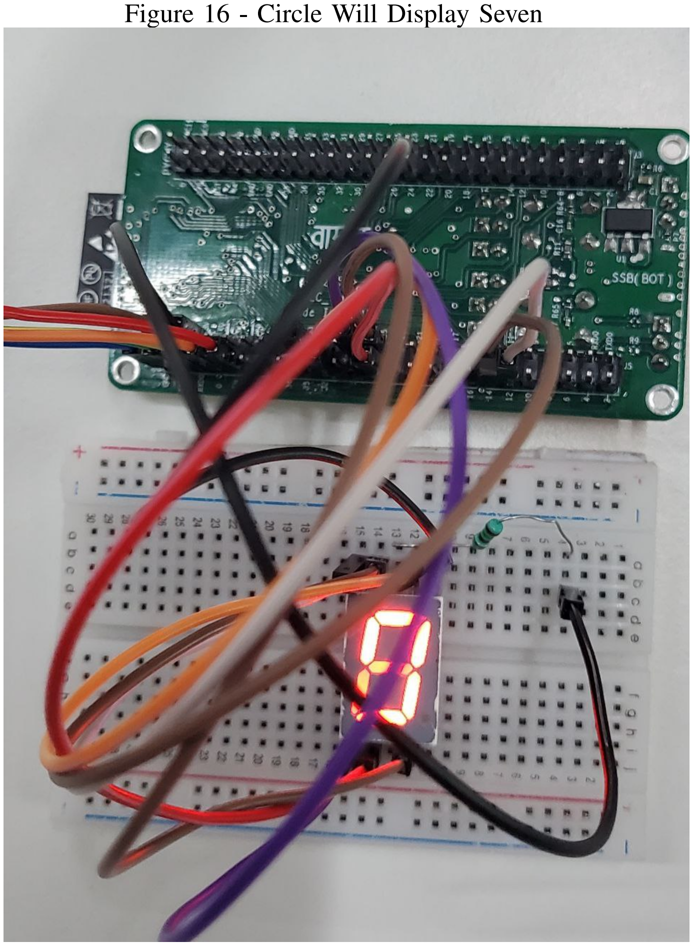


Figure 17 - React Angle Will Display Eight

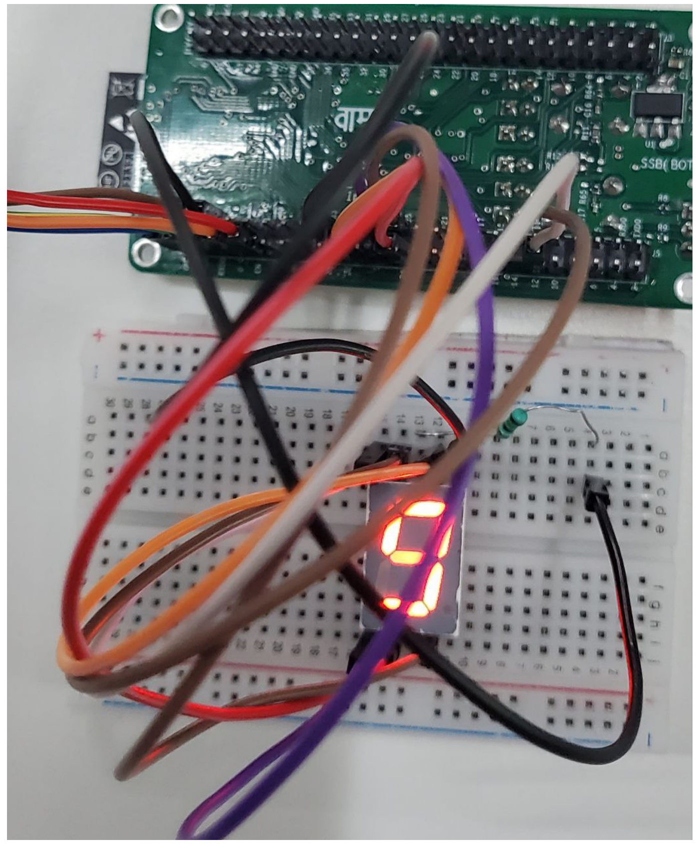


Figure 10 - Triangle Will Display Nine